(19)日本国特許庁(JP)　(12)公開特許公報(A)　(11)特許出願公開番号
特開2004-86545
(P2004-86545A)
(43)公開日　平成16年3月18日(2004.3.18)

(51)Int.Cl.[7]	F I			テーマコード（参考）
G06F 9/44	G06F	9/06	620A	5B076

審査請求 未請求　請求項の数 12　OL　（全 17 頁）

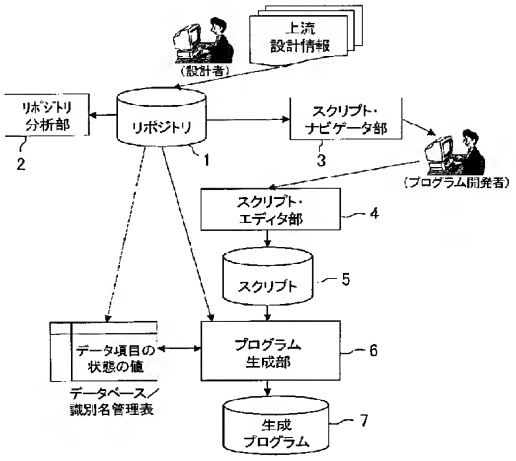
(21)出願番号　特願2002-246441(P2002-246441)
(22)出願日　平成14年8月27日(2002.8.27)

(71)出願人　591030237
日本ユニシス株式会社
東京都江東区豊洲一丁目１番１号
(74)代理人　100059959
弁理士　中村　稔
(74)代理人　100067013
弁理士　大塚　文昭
(74)代理人　100082005
弁理士　熊倉　禎男
(74)代理人　100065189
弁理士　宍戸　嘉一
(74)代理人　100096194
弁理士　竹内　英人
(74)代理人　100074228
弁理士　今城　俊夫

最終頁に続く

(54)【発明の名称】データ項目の状態を管理することでプログラムを自動生成するプログラム開発支援システム

(57)【要約】
【解決手段】プログラム開発支援システムは、設計情報をもとにして、業務領域、データ項目、関数、引数の並び及び／またはエラー処理内容の登録を受け付けて保存する第一の記憶手段と、登録された前記データ項目、関数を用いて簡易言語で記述された業務処理ロジックをスクリプトとして保存する第二の記憶手段と、前記データ項目と前記スクリプトからデータ項目状態管理表を作成し前記データ項目の状態を保存する第三の記憶手段と、前記スクリプトと前記データ項目状態管理表のデータ項目の状態からプログラムを生成するプログラム生成手段と、を備える。
【効果】プログラム開発における生産性の向上、品質の向上、テスト工数及び保守コストの削減ができる。
【選択図】　図１



【特許請求の範囲】
【請求項１】
プログラム開発支援システムにおいて、
設計情報をもとにして、業務領域、データ項目、関数、引数の並び及び／またはエラー処理内容の登録を受け付けて保存する第一の記憶手段と、
登録された前記データ項目、関数を用いて簡易言語で記述された業務処理ロジックをスクリプトとして保存する第二の記憶手段と、
前記データ項目と前記スクリプトからデータ項目状態管理表を作成し前記データ項目の状態を保存する第三の記憶手段と、
前記スクリプトと前記データ項目状態管理表のデータ項目の状態からプログラムを生成するプログラム生成手段と、
を備えることを特徴とするプログラム開発支援システム。
【請求項２】
前記プログラム生成手段は、前記スクリプトに記述された操作の他に、前記データ項目状態管理表のデータ項目の状態および前記第一の記憶手段の保存情報を参照して、データ操作、ワーク領域操作および／またはエラー操作の処理を追加してプログラムを生成するものである請求項１に記載のプログラム開発支援システム。
【請求項３】
前記プログラム生成手段は、生成したプログラムの内容に依存してデータ項目状態管理表のデータ項目の状態を変更するものである請求項１または２に記載のプログラム開発支援システム。
【請求項４】
前記第一の記憶手段には、予約語テーブル、資源テーブル、業務領域テーブル、プログラム部品テーブル、エラー処理テーブルのうちいずれか１つ以上が登録されている請求項１または２または３に記載のプログラム開発支援システム。
【請求項５】
前記第一の記憶手段は、外部記憶装置である請求項１から４のうちのいずれか１項に記載のプログラム開発支援システム。
【請求項６】
前記第二の記憶手段は、外部記憶装置である請求項１から５のうちのいずれか１項に記載のプログラム開発支援システム。
【請求項７】
前記プログラム生成手段によって生成されたプログラムを保存するための第四の記憶手段を更に備える請求項１から６のうちのいずれか１項に記載のプログラム開発支援システム。
【請求項８】
前記第四の記憶手段は、外部記憶装置である請求項７に記載のプログラム開発支援システム。
【請求項９】
前記プログラム生成手段によって生成されるプログラムは、ソースプログラムである請求項１から８のうちのいずれか１項に記載のプログラム開発支援システム。
【請求項１０】
前記プログラム生成手段によって生成されるプログラムは、実行プログラムである請求項１から８のうちのいずれか１項に記載のプログラム開発支援システム。
【請求項１１】
請求項１から１０のうちのいずれか１項に記載のプログラム開発支援システムとしてコンピュータを機能させるためのプログラムを記録したコンピュータ読み取り可能な記録媒体。
【請求項１２】
請求項１から１０のうちのいずれか１項に記載のプログラム開発支援システムとしてコン

ピュータを機能させるためのプログラム。
【発明の詳細な説明】
【０００１】
【発明の属する技術分野】
本発明は、プログラム開発支援システムに関するものであり、特に、コンピュータで処理する業務プログラムの開発に適したプログラム開発支援システムに関するものである。
【０００２】
【従来の技術】
従来、コンピュータで処理する業務プログラムの開発において、分析や設計の成果物、テスト仕様書や結果などの成果物をリポジトリに保存して管理していることが多い。また、データベース定義など設計情報からデータベース生成ＳＱＬやデータベースアクセス用にＣＯＢＯＬ登録集を自動生成する開発支援ツール、プログラム構造を標準化することでスケルトンを生成する開発支援ツール、ひな型からカスタマイズ情報をもとにプログラムを生成する開発支援ツールなどがある。
【０００３】
【特許文献１】
特開２０００－１８１６９３号公報
【特許文献２】
特開２００１－２５６０４９号公報
【特許文献３】
特開２００１－２９６９９６号公報
【０００４】
【発明が解決しようとする課題】
前述したような従来のプログラム開発支援ツールにおいて使用されている開発言語は、種々の目的や領域分野で使用できるようにするため固有の業務処理に依存しない汎用的な言語構成要素を備えている。これら開発言語の構成要素は、コンピュータの基本構造や制御構造が大前提となっていて、コンピュータの構成要素に写像しやすいように設計されている。換言すると、コンピュータに優しい言語であり、それを使う人間には優しくない言語となっており、プログラム開発者である人間の作業負荷を軽減させるという構文規則にはなっていない。また、データベースやコンピュータ資源、再利用する業務部品等も引数を仕様通りに指定して明示的に呼び出さなければならない、等人間の作業負荷は増大している。このような事情からプログラミングには特殊な技能が必要であると考えられていた。さらに、業務機能要件の理解に加えて、開発言語の知識、データベースや通信等といったコンピュータ資源の利用技術、標準化規約等いろいろな情報を熟知しておく必要があり、このことが生産性や品質の向上、期間の短縮、トータルコストの削減といったソフトウエア工学上の諸課題に対する阻害要因となっていた。
【０００５】
この他に、開発言語が汎用的であるといった特性のため、柔軟な言語表現力による不均質や工数の増大、標準化や部品化再利用が強制できずコスト増加、新規開発による品質リスクの増加、ソースコードの増大による品質劣化とテスト工数増大、システム基盤や業務部品再利用等の前提知識を理解するコスト増大、設計情報とソースコードの乖離、等の問題も発生していた。
【０００６】
特に、設計情報とソースコードの乖離の問題について言えば、前述したような従来の開発支援ツールにおいては、基本的には、リポジトリとソースコードとは独立しており、詳細な処理ロジックは生成後のソースコードに対して記述されるため、上流成果物とソースコードとの乖離状態が発生する。上流成果物や設計成果物への更新が既存ソースコードに自動反映されないため、両者を同期させる作業負荷がかかったり、またソースコードは修正するものの設計成果物は放置することになり、文書情報や仕様情報がソースコード上の実態を表していないことになるため機能改良等の際に保守コストが増加することとなる。

【０００７】
特許文献１や特許文献２に開示されている方法は、ソフトウェア部品のひな型から生成したプログラムのデバッグ時等における修正を許しており、これらの問題点を解消できない。特許文献３に開示されている方法も生成したソースコードに後づけで記述を追加するものであり、問題点を解消できない。
【０００８】
この他に、共通プログラム部品をそろえたり、コーディング規約を定めて、標準化を図っている場合もあるが、プログラム開発者の自由度が高く、部品化、標準化を強制する仕組が無いため、プログラム開発者個人の判断で共通プログラムを使用しなかったり、共通プログラムの一部を修正して使用したり、設計者の意図と異なるコーディングをしたりする現象を防止できない。結果として、プログラム品質のばらつきに起因するテスト工数の増加や、その後の機能改良などの保守コストの増加をもたらしている。
【０００９】
本発明の目的は、前述したような従来技術の問題点を解消しうるようなプログラム開発支援システムを提供することである。
【００１０】
【課題を解決するための手段】
本発明の一つの観点によれば、プログラム開発支援システムにおいて、設計情報をもとにして、業務領域、データ項目、関数、引数の並び及び／またはエラー処理内容の登録を受け付けて保存する第一の記憶手段と、登録された前記データ項目、関数を用いて簡易言語で記述された業務処理ロジックをスクリプトとして保存する第二の記憶手段と、前記データ項目と前記スクリプトからデータ項目状態管理表を作成し前記データ項目の状態を保存する第三の記憶手段と、前記スクリプトと前記データ項目状態管理表のデータ項目の状態からプログラムを生成するプログラム生成手段と、を備えることを特徴とするプログラム開発支援システムが提供される。
【００１１】
本発明の一つの実施の形態によれば、前記プログラム生成手段は、前記スクリプトに記述された操作の他に、前記データ項目状態管理表のデータ項目の状態および前記第一の記憶手段の保存情報を参照して、データ操作、ワーク領域操作および／またはエラー操作の処理を追加してプログラムを生成するものである。
【００１２】
本発明の別の実施の形態によれば、前記プログラム生成手段は、生成したプログラムの内容に依存してデータ項目状態管理表のデータ項目の状態を変更するものである。
【００１３】
本発明のさらに別の実施の形態によれば、前記第一の記憶手段には、予約語テーブル、資源テーブル、業務領域テーブル、プログラム部品テーブル、エラー処理テーブルのうちいずれか１つ以上が登録されている。
【００１４】
本発明のさらに別の実施の形態によれば、前記第一の記憶手段は、外部記憶装置である。
【００１５】
本発明のさらに別の実施の形態によれば、前記第二の記憶手段は、外部記憶装置である。
【００１６】
本発明のさらに別の実施の形態によれば、前記プログラム生成手段によって生成されたプログラムを保存するための第四の記憶手段を更に備える。
【００１７】
本発明のさらに別の実施の形態によれば、前記第四の記憶手段は、外部記憶装置である。
【００１８】
本発明のさらに別の実施の形態によれば、前記プログラム生成手段によって生成されるプログラムは、ソースプログラムである。
【００１９】

本発明のさらに別の実施の形態によれば、前記プログラム生成手段によって生成されるプログラムは、実行プログラムである。

【００２０】

本発明の別の観点によれば、前述したようなプログラム開発支援システムとしてコンピュータを機能させるためのプログラムを記録したコンピュータ読み取り可能な記録媒体が提供される。

【００２１】

本発明のさらに別の観点によれば、前述したようなプログラム開発支援システムとしてコンピュータを機能させるためのプログラムが提供される。

【００２２】

【発明の実施の形態】

次に、添付図面に基づいて、本発明の実施の形態および実施例について本発明をより詳細に説明する。

【００２３】

先ず、本発明の実施例について詳述する前に、本発明によるプログラム開発支援システムの概念について説明しておく。

【００２４】

本発明のプログラム開発支援システムの概念は、主として、次の２つの点に主眼をおいている。

（１）設計者によるテーブル指定により業務領域を限定することにより、プログラム開発者の記述した短い簡易言語から、明示的には記述されていないデータベース操作、ワーク領域操作、エラー操作を自動追加してプログラムを生成する。（２）データベース／データ項目名の「状態」を参照してプログラムを自動生成し、そのプログラム生成の結果として「状態」を更新する。この繰り返しにより全体のプログラムを順次自動生成する。

【００２５】

本発明によれば、設計者の意図した内容のプログラム開発を、プログラム開発者に強制的に行わせることができる。プログラム開発者は、業務領域を指定した上で登録済みの識別名を使用して業務プログラムの処理ロジックを簡易言語で記述する。この時使用できるデータ項目名は、設計者があらかじめリポジトリに登録したものだけである。プログラム開発者は、簡易言語による記述を修正できるが、簡易言語から生成されたソースプログラムを修正することはない。

【００２６】

本発明のシステムにおけるプログラム生成部は、単純な置き換え変換でなく、生成した処理内容に対応して、データベースやデータ項目名に値が設定されているか否かの状態変化の情報を内部の表で参照、更新しながら、順次、プログラムを生成して行く。例えば、データベース項目については、データ項目名や主キーの情報がテーブルに登録されているので、スクリプトにおいて、その項目名に参照が発生している場合には、プログラム生成部は、主キーとなるデータ項目がデータ領域に存在し且つ値が入っている事を確認した上で、その値を主キーに設定してデータベースを検索するためのプログラムを自動生成する。同時にワーク領域への転記処理のプログラムを自動生成する。

【００２７】

図１は、本発明の一実施例としてのプログラム開発支援システムのシステム概要を示している。この図１を参照しながら、このプログラム開発支援システムの構成および動作について説明する。先ず、設計者は、上流の設計情報をもとに業務領域で必要となるデータ項目名、関数名の識別名および引数の並びをテーブルに定義し、あらかじめリポジトリ１に登録しておく。このリポジトリ１には、リポジトリ分析部２が関連つけられており、このリポジトリ分析部２は、リポジトリ１の登録内容を分析し、名前の重複や、必要登録事項の漏れなどの登録エラーや修正エラーを報告する。

【００２８】

プログラム開発者は、識別名を用いて、業務処理ロジックのみを簡易言語で記述する。ス

クリプト・ナビゲータ部３は、リポジトリ１に登録されている当該領域で使用可能な識別名などの情報をプログラム開発者へ提供する。

【００２９】

プログラム開発者は、スクリプト・ナビゲータ部３が提供する情報を、スクリプト・エディタ部４へコピー入力することができる。スクリプト・エディタ部４は、プログラム開発者の入力した簡易言語の記述及びスクリプト・ナビゲータ部３が提供する情報を受け付けて、これを編集し外部記憶装置５にスクリプトとして保存する。

【００３０】

プログラム生成部６は、データベース／識別名管理表（データ項目状態管理表）を作成し、この表のデータ項目の状態の値を参照／更新しながら、外部記憶装置５に保存されたスクリプトからＣＯＢＯＬ、Ｃ＋＋などのソースプログラムを生成する。例えば、データベース項目については、データ項目名や主キーの情報がテーブルに登録されているので、スクリプトにおいて、その項目名に参照が発生している場合には、プログラム生成部６は、主キーとなるデータ項目がデータ領域に存在しかつ値が入っている事を確認した上で、その値を主キーに設定してデータベースを検索するためのソースプログラムを自動生成する。生成されたプログラムは、外部記憶装置７に保存される。

【００３１】

本発明のプログラム開発支援システムにおいて、銀行業務を対象にして、ＣＯＢＯＬソースプログラムを生成する具体例について以下説明していく。

【００３２】

この実施例のプログラム開発支援システムのリポジトリ１には、あらかじめ簡易言語の予約語および業務領域の予約語が予約語テーブルとして登録されている。図２に予約語テーブルの登録内容の一例を示している。この図２において、予約語タイプが基本のものについてはプログラム生成部６が構文規則を保持しているものである。予約語タイプが拡張であるものについては、このテーブルで構文解析規則を指定しているものであり、例えば、実行環境ごとに実現方式が変わる予約語である。予約語タイプが設計者定義となっているものは、ある業務領域においてのみ使用される予約語であり、設計者が定義登録することができるものである。

【００３３】

図２の予約語テーブルの生成規則の欄には、後述のプログラム生成部６に対する指示が、簡易言語で記述されている。例えば、予約語“ｉｎｓｅｒｔ”の生成規則の指定において、プログラム部品“ＱＵＥＵＥＰＵＴ”を呼び出すプログラムを生成することが指定されている。なお、プログラム部品については、後述のとおり設計者がリポジトリ１にプログラム部品テーブル（図５参照）として登録する。設計者は、設計情報をもとにして、処理対象の領域情報、データの情報、プログラム部品の情報をあらかじめリポジトリ１に登録しておく。

【００３４】

設計者がリポジトリ１にデータの情報を登録するところの資源テーブルの例を図３に示している。図３の資源テーブルにおける１件目の登録情報では、資源名が“顧客ＤＢ”であるデータベースの資源の物理名が“ｃｕｓｔｏｍｅｒ．ｄｂ”であり、コード生成時の前置詞として“ＣＩＦＤＢ－”を使用し、データ項目名として、顧客番号、氏名、住所、生年月日、電話番号、等を持つことが登録されている。さらに、それぞれのデータ項目のデータ型、桁数が登録されるとともに、主キーが顧客番号であることが登録されている。英字項目名は、言語仕様として日本語変数名を使用できないプログラミング言語（例えばＣ＋＋）において、データ項目名として、ここで指定した値を使用するためのものである。例えばプログラミング言語をＣＯＢＯＬと限定するならば、図３に示すように英字項目名を登録しなくてもよい。

【００３５】

資源テーブルには、キュー、ファイル、メモリー内に保持する内部データなどの情報も同様に登録できる。

【００３６】
設計者がリポジトリ１に処理対象の領域情報を登録するところの業務領域テーブルの例を図４に示している。この図４において、業務領域名が“普通預金入金”であり、使用する資源名が“顧客ＤＢ”と“普通預金ＤＢ”であり、ワーク領域の変数に“ＷＲＫ－”の前置詞をつけることなどが登録されている。この他に、ソースプログラム生成のための生成規則が登録されている。生成規則の欄には、領域における資源の操作手順を前提条件ごと定義することができる。例えば、図４の業務領域テーブルにおける「ルール２」では、“口座番号”に値が設定されていれば、“口座番号”をキーにして“普通預金ＤＢ”を読み込むプログラムを生成し、その後に“顧客番号”をキーにして、“顧客ＤＢ”を読み込むプログラムを生成することが指定されている。プログラム生成部６は、この生成規則を参照してソースプログラムの自動生成を行う。なお、業務領域テーブルにおいて生成規則のルール指定が無い場合であっても、プログラム生成部６は、後述のようにソースプログラムの自動生成を行うことができる。

【００３７】
設計者がリポジトリ１にプログラム部品の情報を登録するところのプログラム部品テーブルの例を図５に示している。この図５において、プログラム部品名、戻り値、引数の並び、資源名等が登録されている。図５のプログラム部品テーブルにおける「引数の並び」の値として、＜・・・＞の形式で指定されているものは可変名称であり、プログラム生成部６が判断して変数名を生成するものである。＜・・・＞の形式でないものは、図３の資源テーブルにおいて、データ項目名として登録されている必要がある。この場合、図５のプログラム部品テーブルの資源名の値と図３の資源テーブルの資源名の値も一致していなければならない。プログラム生成部６は、このプログラム部品テーブルを参照してプログラムを生成するので、プログラム開発者がスクリプト記述において引数の記述を省略することができる。この場合、プログラム生成部６は、引数を補充してプログラムを生成する。なお、図５のプログラム部品テーブルの資源名の値が＜ｄａｔａｂａｓｅ＞となっている場合は、資源テーブル（図３参照）において種別の値がデータベースとして登録されているいずれの資源であってもよいことを示している。なお、図５のプログラム部品テーブルにおいてプログラム部品名の値が“照会”であるものが複数登録されているが、資源名の値が異なるので、それぞれ別の部品であることがわかる。また、図５のプログラム部品テーブルの例では、戻り値の欄に＜ｅｒｒｏｒ＞が指定されているが、例えば戻り値を持たないＣＯＢＯＬグラムを生成する場合には、プログラム生成部６は、引数の最後に戻り値を追加した形式のソースプログラムを生成する。

【００３８】
設計者がリポジトリ１に登録するところのエラー処理テーブルの例を図６に示している。この図６では、エラー処理名“普通預金ＤＢロード後”において、データ項目の口座番号についてのエラー処理として、副プログラム“口座番号検査”を呼び、結果のエラーステータスが０でなければ、副プログラム“ＤＢ不正”を呼ぶプログラムを生成することを指定している。顧客番号、残高についてのエラー処理も同じように指定されている。

【００３９】
スクリプト・ナビゲータ部３が、リポジトリ１の登録情報を提供し支援する環境で、スクリプト・エディタ部４が、プログラム開発者の入力した簡易言語の記述を受け入れて、編集し出力したスクリプトの一例を図７に示している。この図７のスクリプトでは、業務領域“普通預金入金”において、口座番号を受け付けて、顧客番号、口座番号、氏名、残高を出力する処理を指定している。これらのエラー処理についてはプログラム開発者が記述することはなく、リポジトリ１のテーブルを参照して、プログラム生成部６が自動生成する。このようにすることにより、プログラム開発者によりエラー処理内容が異なるというような、品質のばらつきを防止できる。

【００４０】
プログラム生成部６がスクリプトを読み込んでリポジトリ１を参照しながらＣＯＢＯＬソースコードを生成する処理の流れを図８に示している。この図８の処理の流れに沿って以

下説明する。

【００４１】

先ず、この処理の流れについて説明する前に、本発明のシステムにおけるプログラム生成部６の動作の特徴部分を理解し易くするため、従来のものと対比して説明しておく。図９は、図７のスクリプトの例について、このような本発明の特徴部分を従来のものと対比して図式的に説明している図である。この図９に図式的に示すように、従来方式においては、図７のスクリプトだけのプログラムでは、３行目のｐｒｉｎｔ文により、プリントアウトされるものは、０　１２３４５６７￥０　となってしまうことが予想される。しかし、図７のスクリプトにおいて、”ａｃｃｅｐｔ口座番号”と”ｐｒｉｎｔ顧客番号、口座番号、氏名、残高”との間に、４行のデータベース操作文、すなわち、”ｏｐｅｎ普通預金ＤＢ”、”ｆｉｎｄ普通預金レコードｕｓｉｎｇ口座番号”、”ｏｐｅｎ顧客ＤＢ”、”ｆｉｎｄ顧客レコードｕｓｉｎｇ顧客番号”が追加されると、ｐｒｉｎｔ文により、プリントアウトされるものは、５８７４１２３４５６７　豊洲太郎　￥１１，０００，０００　という正しい結果が期待できる。従来方式においては、これら４行は、必要となる変数がどこに保存されているかをプログラマが知っているので、それらを正しい順序で、自分のデータ領域へ取り出していたのであるが、本発明では、必要となる変数を記述しなくても、正しい順序で、自分のデータ領域へ取り出す部分を自動生成するのである。

【００４２】

このような自動生成を行うため、先ず、プログラム生成部６は、ステップＳ１１０において、外部記憶装置５に保存されたスクリプトの先頭のｕｓｅｃａｓｅ文を読み込む。続いて、プログラム生成部６は、読み込んだスクリプトの、ｕｓｅｃａｓｅ文を分析して、リポジトリ１から関連するテーブルを読み込む。例えば、図７のスクリプトの場合に、プログラム生成部６は、ｕｓｅｃａｓｅ文から業務領域が“普通預金入金”であることを知り、業務領域テーブルの該当する個所を読み込む。続いてプログラム生成部６は、例えば、図４の業務領域テーブルの「使用する資源名」の値をもとにして、使用する資源が“顧客ＤＢ”と“普通預金ＤＢ”であることを知り、資源テーブルの該当する個所を読み込む。

【００４３】

さらに、プログラム生成部６は、ステップＳ１３０にて、読み込んだテーブル情報をもとにして、データベースや識別名の状態変化を記憶するためのデータベース／識別名管理表を作成する。

【００４４】

図１０は、このようにしてプログラム生成部６がスクリプトに基づいてリポジトリ１を参照しつつデータベース／識別名管理表を作成することによりプログラムを自動生成していく手順を分かり易くするため、リポジトリ１とプログラム生成部６との関係を図式化してまとめて示す図である。この図１０に示されるように、プログラム生成部６は、読み込んだスクリプトにしたがったプログラムで使用するデータ項目である、普通預金ＤＢデータ項目、顧客ＤＢデータ項目等に関連するテーブルを、リポジトリ１から読み込み、読み込んだテーブル情報をもとにしてデータベース／識別名管理表を作成し、バッファ部６Ａに記憶させる。このようにしてプログラム生成部６のバッファ部６Ａに記憶されるデータベース／識別名管理表の例を図１１に示している。

【００４５】

この図１１のデータベース／識別名管理表において、項目名“顧客番号”は、定義されている資源名“顧客ＤＢ”と“普通預金ＤＢ”に存在する事を示す。プログラム生成部６は、データベース／識別名管理表の作成時点では「状態」の値を“未オープン”や“未初期化”としておく。この後、プログラム生成部６は、次のような手順により、データベース／識別名管理表の「状態」の値を参照及び更新しながら、プログラム生成を進めて行くのである。

【００４６】

図８の流れ図に戻って、プログラム生成部６は、ステップＳ１４０において、ステップＳ１２０で読み込んだテーブルをもとにして、スクリプトの構文検査を行う。この場合にお

いて、プログラム生成部６は、スクリプトに致命的な構文エラーがあり、ソースプログラムを生成できない場合には、ステップＳ１５０にて、構文エラー情報を出力し、処理を終了する。

【００４７】

一方、プログラム生成部６は、構文検査ＯＫの場合には、ステップＳ１６０にて、スクリプトを分析し、読み込んだテーブル及びデータベース／識別名管理表の「状態」の値を参照して、ソースプログラムを生成する。例えば、図７のスクリプトのｕｓｅｃａｓｅ文と業務領域テーブル（図４参照）、資源テーブル（図３参照）からＣＯＢＯＬソースプログラムのデータ部（図１２参照）、処理部（図１３参照）を生成する。また、ａｃｃｅｐｔ文、ｐｒｉｎｔ文と各種テーブルからＣＯＢＯＬソースプログラムの処理部（図１４参照）を生成する。

【００４８】

プログラム生成部６は、例えば、図７のスクリプトのａｃｃｅｐｔ文に対応するＣＯＢＯＬソースプログラムコードを生成すると、ステップＳ１７０において、データベース／識別名管理表の、口座番号の「状態」の値を“外部入力済み”に更新する（図１５参照）。続いて、プログラム生成部６は、ステップＳ１８０において次のスクリプトを読み込み、終了でないのでステップＳ１４０、Ｓ１６０およびＳ１７０に戻り、図７のスクリプトのｐｒｉｎｔ文に対応するＣＯＢＯＬソースプログラムコードを生成する。プログラム生成部６は、普通預金ＤＢの読み込みの処理の部分を生成すると、データベース／識別名管理表の、普通預金ＤＢの顧客番号、口座番号、残高の「状態」の値を“ＤＢ読込済み”に更新する（図１６参照）。

【００４９】

最後に、プログラム生成部６は、コード生成終了に応じて、スクリプトの文の解析を終えて、ステップＳ１９０において必要に応じ、使用しないデータ部のソースプログラム部分の削除等の最適化を行い、最終的に確定したソースプログラムを外部記憶装置７に書き出す。

【００５０】

以降、プログラム生成部６が、ステップＳ１４０、Ｓ１６０およびＳ１７０において、読み込んだテーブルの参照とデータベース／識別名管理表の参照、更新を行いながらソースプログラムを生成する処理について、更に詳しく説明する。プログラム生成部６が、図７に示すスクリプトの、ｕｓｅｃａｓｅ文とａｃｃｅｐｔ文を処理し終わり、ｐｒｉｎｔ文の処理を開始するところから説明する。

【００５１】

図４の業務領域テーブルの生成規則にルールの指定がある場合

このとき、データベース／識別名管理表は、図１５のようになっている。プログラム生成部６は、図１５から、普通預金ＤＢの主キーである口座番号の状態が“外部入力済み”である事を知り、図４の業務領域テーブルの生成規則の「ルール２」が成り立つことを知る。「ルール２」をもとに、普通預金ＤＢのロードが必要なことを知る。プログラム生成部６は、プログラム部品テーブル（図５参照）を参照して、普通預金ＤＢのオープンの副プログラム呼び出しのソースプログラムを生成する（図１４参照）。続いて、図１５のデータベース／識別名管理表における口座番号の状態が“外部入力済み”であるので、口座番号をキーとしたセレクトの副プログラム呼び出しのソースプログラムを生成する（図１４参照）。そして、プログラム生成部６は、これらの普通預金ＤＢのオープンとセレクトのソースプログラム生成を反映して、データベース／識別名管理表の該当する項目名の状態を図１６に示すように“オープン済み”、“ＤＢ読込済み”に更新する。

【００５２】

続いて、プログラム生成部６は、図６のエラー処理テーブルを参照して、普通預金ＤＢのエラー処理時の欄に“ロード後”が指定されているのをみて、普通預金ＤＢのロード後には、エラー処理名が“普通預金ＤＢロード後”であるエラー処理が必要なことを知り、データ項目の口座番号、顧客番号、残高のエラー処理を行うソースプログラムを生成する（

図１７参照）。

【００５３】

この後、プログラム生成部６は、図４の「ルール２」に従い、顧客ＤＢのロードが必要なことを知り、顧客ＤＢのオープンの副プログラム呼び出しのソースプログラムを生成する（図示せず）。更に、図１６のデータベース／識別名管理表における顧客番号を参照し、「定義されている資源名」が“普通預金ＤＢ”である顧客番号の状態が“ＤＢ読込済み”であることを知り、この値をキーとして、顧客ＤＢの読み込みの処理の部分を生成する（図示せず）。続いて、プログラム生成部６は、図６のエラー処理テーブルを参照して、顧客ＤＢのエラー処理時の欄に“ロード後”が指定されているのをみてエラー処理名“顧客ＤＢロード後”のエラー処理が必要なことを知り、エラー処理を行うソースプログラムを生成する（図示せず）。

【００５４】

このように、プログラム生成部６は、データベース／識別名管理表の項目名の状態の値を参照、更新しながら、ソースプログラム生成を続ける。

【００５５】

図４の業務領域テーブルの生成規則にルールの指定が無い場合

プログラム生成部６は、スクリプトの　ｐｒｉｎｔ文を解析して、顧客番号、口座番号、氏名、残高の値が必要であることを知る。図３の資源テーブルを参照して、データベース“顧客ＤＢ”、“普通預金ＤＢ”にこれらの値があることを知る。図１５のデータベース／識別名管理表から、顧客ＤＢの主キーである顧客番号の値が“未初期化”であることを知り、顧客ＤＢがロードされていないこと及びロードできないことがわかる。同じく、図１５のデータベース／識別名管理表から、普通預金ＤＢの主キーである口座番号の状態が“外部入力済み”であることを知り、普通預金ＤＢがロードされていないこととおよびロード可能であることがわかる。プログラム生成部６は、プログラム部品テーブル（図５参照）を参照して、普通預金ＤＢのオープンの副プログラム呼び出しのソースプログラムを生成する（図１２参照）。続いて、図１５のデータベース／識別名管理表における口座番号の状態が“外部入力済み”であるので、口座番号をキーとしたセレクトの副プログラム呼び出しのソースプログラムを生成する（図１４参照）。そして、プログラム生成部６は、これらの普通預金ＤＢのオープンとセレクトのソースプログラム生成を反映して、データベース／識別名管理表の該当する項目名の状態を図１６に示すように“オープン済み”、“ＤＢ読込済み”更新する。

【００５６】

続いて、プログラム生成部６は、図６のエラー処理テーブルを参照して、普通預金ＤＢのエラー処理時の欄に“ロード後”が指定されているのをみて、普通預金ＤＢのロード後には、エラー処理名が“普通預金ＤＢロード後”であるエラー処理が必要なことを知り、データ項目の口座番号、顧客番号、残高のエラー処理を行うソースプログラムを生成する（図１７参照）。

【００５７】

この後、プログラム生成部６は、プログラム部品テーブル（図５参照）を参照して、顧客ＤＢのオープンの副プログラム呼び出しのソースプログラムを生成する（図示せず）。更に、図１６のデータベース／識別名管理表における顧客番号を参照し、「定義されている資源名」が“普通預金ＤＢ”である顧客番号の状態が“ＤＢ読込済み”であることを知り、この値をキーとして、顧客ＤＢの読み込みの処理の部分を生成する（図示せず）。続いて、プログラム生成部６は、図６のエラー処理テーブルを参照して、顧客ＤＢのエラー処理時の欄に“ロード後”が指定されているのをみてエラー処理名“顧客ＤＢロード後”のエラー処理が必要なことを知り、エラー処理を行うソースプログラムを生成する（図示せず）。

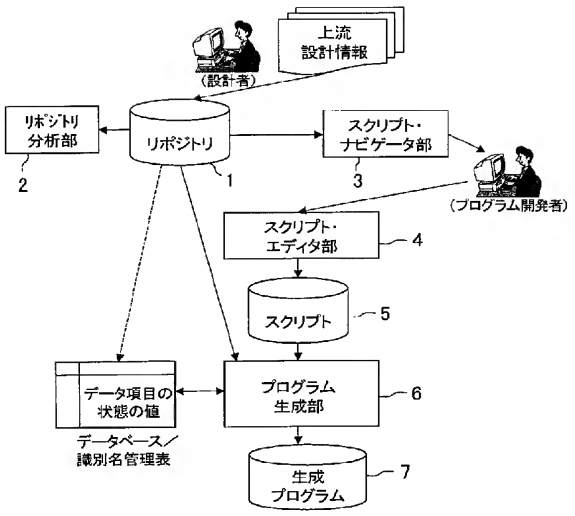
【００５８】

このように、プログラム生成部６は、データベース／識別名管理表の状態の値を参照、更新しながら、ソースプログラム生成を続ける。

【００５９】
前述した実施例の説明では、プログラム生成部６がＣＯＢＯＬソースプログラムを生成する例についてであったが、プログラム生成部６は、Ｃ＋＋などの他のプログラミング言語のソースプログラムを生成するようにしてもよい。
【００６０】
また、Ｓ１９０において、プログラム生成部６は、ソースプログラムをコンパイルして、さらに実行プログラムを生成して、外部記憶装置に書き出すようにしてもよい。
【００６１】
さらにまた、プログラム生成部６は、ステップＳ１６０においてソースプログラムではなく中間テーブルを生成し、ステップＳ１９０において、中間テーブルから、ＣＯＢＯＬやＣ＋＋などのソースプログラムあるいは、実行プログラムを生成して書き出してもよい。
【００６２】
【発明の効果】
本発明によれば、プログラム開発者は、データ項目名と関数名だけを記述するだけになるため、見かけのソースコード数が削減され可読性が高まり、生産性が向上する。また、ソースコード数が減少するため比例してテスト工数が削減できる。
【００６３】
データ項目の値を得るためのデータベース参照処理等が自動化されるため、あたかもデータベースを使用していないかのような操作が可能となり、生産性が向上する。
【００６４】
データ項目名等識別名が設計時に定義されるため、用語の標準化が機構的に実現できる。
【００６５】
上流設計者においてデータベースや再利用する部品が定義され、プログラム開発者がその定義を回避する手段を持たないため部品化再利用が強制される。この部品再利用の継続により部品品質が向上する。
【００６６】
データベースの保守処理やバッファ間のデータ転記処理が自動生成され、データ項目追加やデータベース仕様の再編成があってもスクリプトには影響や修正作業が発生しないため、コストおよびデグレード（修正に伴う他の不具合の発生）リスクを抑えることができる。
【００６７】
業務処理部品のインタフェースや引数の並びもリポジトリで設計登録しておくため、呼出処理が自動生成できる。この結果、引数の追加等のような仕様変更があっても、スクリプトには影響や修正作業が発生しないため、コストおよびデグレードリスクを抑えることができる。
【００６８】
上流設計とプログラム開発との成果物分離と役割分担が強制される。これにより、上流工程における下流工程の自由度を制約することで要求の実現を確実化する。
【００６９】
ソースコードがスクリプトとリポジトリの設計情報から生成されるため、設計情報とソースコードの内容が完全に同期する。これによってソースコードではなく設計仕様やスクリプトだけで機能改善を行うことが可能となり、生産性や保守性が向上する。
【図面の簡単な説明】
【図１】本発明による一実施例としてのプログラム開発支援システムのシステム概要を示す図である。
【図２】予約語テーブルの登録内容の一例を示す図である。
【図３】資源テーブルの一例を示す図である。
【図４】業務領域テーブルの一例を示す図である。
【図５】プログラム部品テーブルの一例を示す図である。
【図６】エラー処理テーブルの一例を示す図である。

【図７】スクリプトの一例を示す図である。
【図８】プログラム生成部がプログラムを生成する処理の流れを示す図である。
【図９】図７のスクリプトの例について本発明の特徴部分を従来のものと対比して図式的に説明している図である。
【図１０】リポジトリとプログラム生成部との関係を図式化してまとめて示す図である。
【図１１】プログラム生成部が作成するデータベース／識別名管理表の一例を示す図である。
【図１２】プログラム生成部が作成するＣＯＢＯＬソースプログラムのデータ部の一例を示す図である。
【図１３】プログラム生成部が作成するＣＯＢＯＬソースプログラムの処理部の一例を示す図である。
【図１４】プログラム生成部が作成するＣＯＢＯＬソースプログラムの処理部の別の例を示す図である。
【図１５】プログラム生成部が作成するデータベース／識別名管理表の別の例を示す図である。
【図１６】プログラム生成部が作成するデータベース／識別名管理表のさらに別の例を示す図である。
【図１７】プログラム生成部が作成するＣＯＢＯＬソースプログラムの処理部のさらに別の例を示す図である。
【符号の説明】
１　リポジトリ
２　リポジトリ分析部
３　スクリプト・ナビゲータ部
４　スクリプト・エディタ部
５　外部記憶装置
６　プログラム生成部
６Ａ　バッファ部
７　外部記憶装置

【図１】



【図２】

予約語テーブル

予約語	予約語タイプ	構文解析規則	生成規則
if	基本		
while	基本		
foreach	基本		
usecase	基本		
・・・	・・・		～～
accept	拡張	accept 〈左辺値〉[*]	～～
print	拡張	print 〈右辺値〉[〈定数〉]*	～～
insert	拡張	insert 〈キュー〉 〈挿入データ〉	QUEUEPUT(オンラインQ, 送信データ)
remove	拡張	remove 〈キュー〉 〈取り出しデータ〉	～～
・・・	・・・	・・・	・・・
口銀送信	設計者定義	口銀送信 〈送信情報〉 〈送信データ〉	～～
ログ書込	設計者定義	ログ書込 〈ログ情報〉	～～
・・・	・・・	・・・	・・・

【図３】

資源テーブルの例

種別	資源名	資源の物理名	前置詞	データ項目名	英字項目名	データ型	桁数	主キー	・・・
データベース	顧客DB	customer.db	CTFDB-	顧客番号		文字列型	10	○	
				氏名		文字列型	20		
				住所		文字列型	60		
				生年月日		日付型	8		
				電話番号		文字列型	12		
				・・・		・・・	・・		
データベース	普通預金DB	ftm.db	FTMDB-	口座番号		文字列型	7	○	
				顧客番号		文字列型	10		
				残高		整数型	15		
				・・・		・・・	・・		
キュー	オンラインQ	～～		送信データ		文字列型	可変		
ファイル	～～	～～		～～		～～	～～		
内部データ	～～	～～		～～		～～	～～		

【図４】

業務領域テーブルの例

業務領域名	普通預金入金
使用する資源名	顧客DB
	普通預金DB
変数宣言	
ワーク領域	WRK-
～～	～～
生成規則	
ルール1	condition 顧客番号 load 顧客DB using 顧客番号 load 普通預金DB using 顧客番号
ルール2	condition 口座番号 load 普通預金DB using 口座番号 load 顧客DB using 顧客番号
～～	

【図５】

プログラム部品テーブルの例

No.	プログラム部品名	戻り値	引数の並び	資源名	・・・
1	DBOPEN	<error>	<name>	<database>	～
2	DBSQL	<error>	<name>, <sql>, <record>	<database>	～
…	・・・	・・・	・・・	・・・	～
10	QUEUEPUT	<error>	<data>	<queue>	～
11	照会	<error>	口座番号，顧客番号，残高	普通預金	～
12	出金	<error>	口座番号，出金額，残高	普通預金	～
…	・・・	・・・	・・・		～
21	照会	<error>	顧客番号，氏名，住所，電話番号	顧客	～
…	・・・	・・・	・・・	・・・	・・・

【図６】

エラー処理テーブルの例

対象データベース名	エラー処理時	エラー処理名	データ項目 口座番号	顧客番号	残高	…
普通預金DB	ロード後	普通預金DBロード後	口座番号検査() if エラーステータス != 0 then DB不正()	顧客番号検査() if エラーステータス != 0 then DB不正()	残高検査() if エラーステータス != 0 then DB不正()	
	update前	普通預金DB.update前	～～	～～	～～	
	create前	普通預金DB.create前	～～	～～	～～	
	delete前	普通預金DB.delete前	～～	～～	～～	
	・・・	・・・				

対象データベース名	エラー処理時	エラー処理名	データ項目 顧客番号	氏名	住所	…
顧客DB	ロード後	顧客DBロード後	顧客番号検査C() if エラーステータス != 0 then DB不正()	～～	～～	
	update前	顧客DB.update前	～～	～～	～～	
	create前	顧客DB.create前	～～	～～	～～	
	delete前	顧客DB.delete前	～～	～～	～～	
	・・・	・・・				

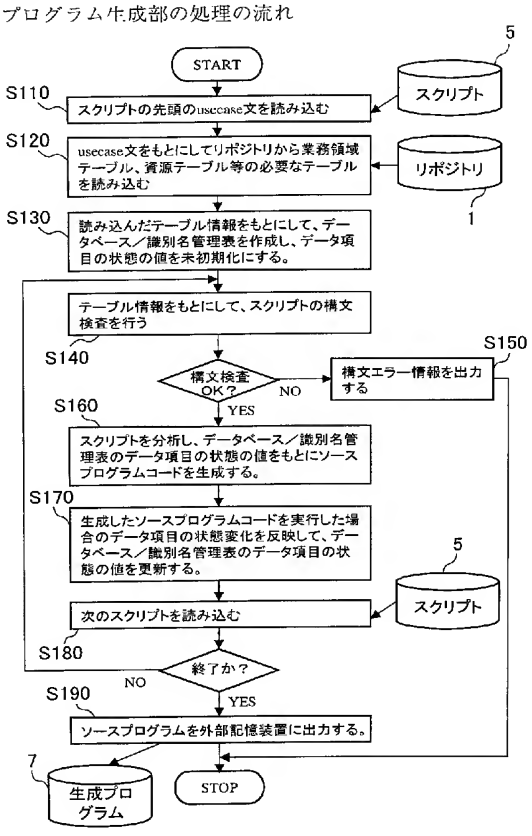
【図７】

スクリプトの例

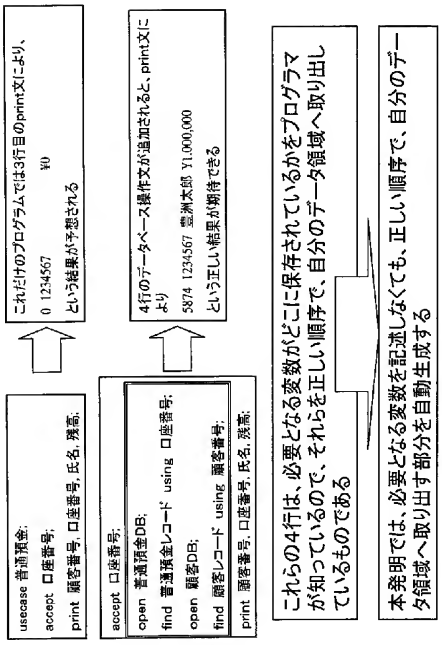
```
usecase 普通預金入金；
accept　口座番号；
print 顧客番号、口座番号、氏名、残高；
```

【図８】

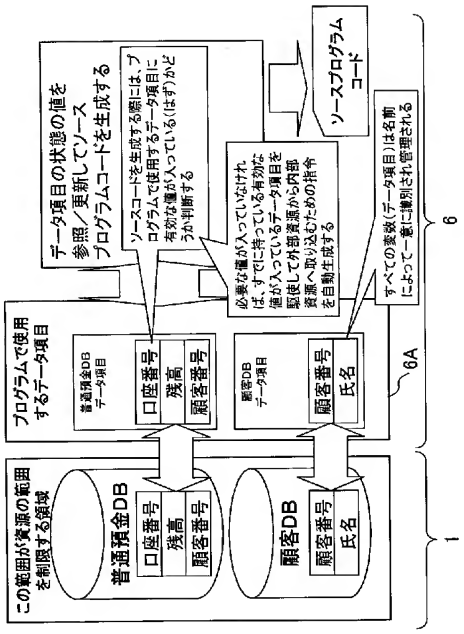
プログラム生成部の処理の流れ



【図９】



【図１０】



【図１１】

データベース／識別名管理表(その１)

項目名	種別	定義されている資源名	状態
顧客ＤＢ	データベース		未オープン
普通預金ＤＢ	データベース		未オープン
・・・	・・・	・・・	・・・
顧客番号	データ項目名	顧客ＤＢ	未初期化
		普通預金ＤＢ	未初期化
氏名	データ項目名	顧客ＤＢ	未初期化
住所	データ項目名	顧客ＤＢ	未初期化
生年月日	データ項目名	顧客ＤＢ	未初期化
電話番号	データ項目名	顧客ＤＢ	未初期化
口座番号	データ項目名	普通預金ＤＢ	未初期化
残高	データ項目名	普通預金ＤＢ	未初期化
・・・	・・・	・・・	・・・

【図１２】

ＣＯＢＯＬソースプログラム(データ部)

```
AUTO  *ユースケース名                    普通預金入金
AUTO  *ユースケース識別名                FTDEPOSIT
AUTO   IDENTIFICATION                    DIVISION.
AUTO   PROGRAM-ID.                       FTDEPOSIT.
AUTO   ENVIRONMENT                       DIVISION.
AUTO   CONFIGURATION                     SECTION.
AUTO   DATA                              DIVISION.
AUTO   WORKING-STORAGE                   SECTION.
AUTO  ***-----------------------------------------------
AUTO  *** usecase 普通預金入金
AUTO  ***-----------------------------------------------
AUTO  ***データベース定義
AUTO   01  顧客ＤＢレコード定義.
AUTO         03  ＣＩＦＤＢ－顧客番号      PIC  X(10).
AUTO         03  ＣＩＦＤＢ－氏名          PIC  X(20).
AUTO         03  ＣＩＦＤＢ－住所          PIC  X(60).
AUTO         03  ＣＩＦＤＢ－生年月日      PIC  X(08).
AUTO         03  ＣＩＦＤＢ－電話番号      PIC  X(12).
AUTO   01  普通預金ＤＢレコード定義.
AUTO         03  ＦＴＭＤＢ－口座番号      PIC  X(07).
AUTO         03  ＦＴＭＤＢ－顧客番号      PIC  X(10).
AUTO         03  ＦＴＭＤＢ－残高          PIC  X(15).
AUTO
AUTO  ***内部処理用ワーク領域
AUTO   01  顧客ＤＢ定義対応ワーク.
AUTO         03  ＷＲＫ－顧客番号          PIC  X(10).
AUTO         03  ＷＲＫ－氏名              PIC  X(20).
              ～～～～
```

【図１３】

ＣＯＢＯＬソースプログラム(処理部その１)

```
AUTO   PROCEDURE                    DIVISION.
AUTO   メインコントロール            SECTION.
AUTO   メインコントロールＳ.
AUTO       PERFORM  メイン初期処理
AUTO       PERFORM  メイン処理
AUTO       PERFORM  メイン後処理
AUTO       EXIT  PROGRAM.
AUTO   メインコントロールＥ.
AUTO       EXIT.

AUTO   メイン初期処理               SECTION.
AUTO   メイン初期処理Ｓ.
AUTO   作業領域の初期化.
AUTO       MOVE  SPACE              TO  顧客ＤＢレコード定義.
AUTO       MOVE  SPACE              TO  普通預金ＤＢレコード定義.
AUTO       MOVE  SPACE              TO  顧客ＤＢ定義対応ワーク.
AUTO       MOVE  SPACE              TO  普通預金ＤＢ定義対応ワーク.
AUTO       MOVE  SPACE              TO  ＳＣＥＲＲ.
           ～～～～
           ～～～～
AUTO   メイン初期処理Ｅ.
AUTO       EXIT.

AUTO   メイン処理                   SECTION.
AUTO   メイン処理Ｓ.
           ～～～～
```

【図１４】

ＣＯＢＯＬソースプログラム(処理部その２)

```
AUTO ***=============================================
AUTO *** accept 口座
AUTO ***=============================================
AUTO    ACCEPT WRＫ−口座番号.
AUTO ***=============================================
AUTO *** print 氏名,口座番号,顧客番号,残高.
AUTO ***=============================================
AUTO ***普通預金ＤＢデータベース　ＯＰＥＮ
AUTO    CALL "DBOPEN" USING 普通預金ＤＢＮＡＭＥ
AUTO                ＳＣＥＲＲ
AUTO    IF ＳＣＥＲＲ エラーステータス NOT = ZERO
AUTO     THEN
AUTO      GO TO メインＥ
AUTO     END-IF
AUTO ***普通預金ＤＢデータベース　ＳＥＬＥＣＴ
AUTO    MOVE "SELECT 口座番号,顧客番号,残高 FROM 普通預金ＤＢ WHERE 口座番号=" 
AUTO       ＷＲＫ−口座番号
AUTO      TO 普通預金ＤＢＳＱＬ
AUTO    CALL "DBSQL" USING 普通預金ＤＢＮＡＭＥ
AUTO                普通預金ＤＢＳＱＬ
AUTO                普通預金ＤＢレコード定義Ｌ
AUTO                ＳＣＥＲＲ
AUTO    IF ＳＣＥＲＲ−エラーステータス NOT = ZERO
AUTO     THEN
AUTO      GO TO メインＥ
AUTO     END-IF
           ～～～～
```

【図１５】

データベース／識別名管理表(その２)

項目名	種別	定義されている資源名	状態
顧客ＤＢ	データベース		未オープン
普通預金ＤＢ	データベース		未オープン
・・・	・・・	・・・	・・・
顧客番号	データ項目名	顧客ＤＢ	未初期化
		普通預金ＤＢ	未初期化
氏名	データ項目名	顧客ＤＢ	未初期化
住所	データ項目名	顧客ＤＢ	未初期化
生年月日	データ項目名	顧客ＤＢ	未初期化
電話番号	データ項目名	顧客ＤＢ	未初期化
口座番号	データ項目名	普通預金ＤＢ	外部入力済み
残高	データ項目名	普通預金ＤＢ	未初期化
・・・	・・・	・・・	・・・

【図１６】

データベース／識別名管理表(その３)

項目名	種別	定義されている資源名	状態
顧客ＤＢ	データベース		未オープン
普通預金ＤＢ	データベース		オープン済み
・・・	・・・	・・・	・・・
顧客番号	データ項目名	顧客ＤＢ	未初期化
		普通預金ＤＢ	ＤＢ読込済み
氏名	データ項目名	顧客ＤＢ	未初期化
住所	データ項目名	顧客ＤＢ	未初期化
生年月日	データ項目名	顧客ＤＢ	未初期化
電話番号	データ項目名	顧客ＤＢ	未初期化
口座番号	データ項目名	普通預金ＤＢ	ＤＢ読込済み
残高	データ項目名	普通預金ＤＢ	ＤＢ読込済み
・・・	・・・	・・・	・・・

【図１７】

ＣＯＢＯＬソースプログラム(処理部その２)

```
AUTO ***普通預金ＤＢ項目　検査処理
AUTO    PERFORM 普通預金ＤＢロード後
AUTO    IF ＳＣＥＲＲ−エラーステータス NOT = ZERO
AUTO     THEN
AUTO      GO TO メインＥ
AUTO     END-IF
AUTO ・・・・・・・・・・・・
AUTO ・・・・・・・・・・・・
AUTO ************************************************************
AUTO 普通預金ＤＢロード後　　SECTION.
AUTO ************************************************************
AUTO 普通預金ＤＢロード後Ｓ.
AUTO 普通預金ＤＢロード後処理.
AUTO    CALL "口座番号検査" USING ＦＴＭＤＢ−口座番号Ｌ
AUTO                ＳＣＥＲＲ
AUTO    IF ＳＣＥＲＲ−エラーステータス NOT = ZERO
AUTO     THEN
AUTO      PERFORM ＤＢ不正
AUTO     END-IF
AUTO    CALL "顧客番号検査" USING ＦＴＭＤＢ−顧客番号Ｌ
AUTO                ＳＣＥＲＲ
AUTO    IF ＳＣＥＲＲ−エラーステータス NOT = ZERO
AUTO     THEN
AUTO      PERFORM ＤＢ不正
AUTO     END-IF
           ～～～～
```

フロントページの続き

(74)代理人　100084009
弁理士　小川　信夫
(74)代理人　100082821
弁理士　村社　厚夫
(74)代理人　100086771
弁理士　西島　孝喜
(74)代理人　100084663
弁理士　箱田　篤
(72)発明者　石田　政海
東京都江東区豊洲１丁目１番１号　日本ユニシス株式会社内
Ｆターム(参考) 5B076 DD02　DD09　DD10　EC07

DERWENT-ACC-NO: 2004-288701

DERWENT-WEEK: 200427

TITLE: Source program development support system stores data item and transaction processing logic as script for producing data item state management table based on which source program is generated

INVENTOR: ISHIDA M

PATENT-ASSIGNEE: NIPPON UNISYS LTD[BURS]

PRIORITY-DATA: 2002JP-246441 (August 27, 2002)

PATENT-FAMILY:

PUB-NO	PUB-DATE	LANGUAGE
JP 2004086545 A	March 18, 2004	JA

APPLICATION-DATA:

PUB-NO	APPL-DESCRIPTOR	APPL-NO	APPL-DATE
JP2004086545A	N/A	2002JP-246441	August 27, 2002

INT-CL-CURRENT:

TYPE	IPC DATE
CIPP	G06F9/44 20060101

ABSTRACTED-PUB-NO: JP 2004086545 A

BASIC-ABSTRACT:

NOVELTY - An external storage device (5) stores information about work area, data item, function, argument and content of error handling, based on design information. A memory stores data item and transaction processing logic as a script. Another external storage device (7) stores data item state management table produced using data item and script. A source program is generated based on the management table.

DESCRIPTION - INDEPENDENT CLAIMS are also included for the following:

(1) program development support program; and

(2) computer readable recorded medium for storing program development support program.

USE - For supporting development of source program such as C++ and COBOL, using computer.

ADVANTAGE - The source code number is reduced, since the program developer describes only data item name and function name, therefore the test man hour is reduced, and the productivity is improved.

DESCRIPTION OF DRAWING(S) - The figure shows the profile of the work program development support system. (Drawing

includes non-English language text).

repository (1)

repository analysis unit (2)

external storage devices (5,7)

program generation unit (6)

CHOSEN-DRAWING: Dwg.1/17

TITLE-TERMS: SOURCE PROGRAM DEVELOP SUPPORT SYSTEM STORAGE DATA ITEM TRANSACTION PROCESS LOGIC SCRIPT PRODUCE STATE MANAGEMENT TABLE BASED GENERATE

DERWENT-CLASS: T01

EPI-CODES: T01-G08A; T01-J20A; T01-S03;

SECONDARY-ACC-NO:
Non-CPI Secondary Accession Numbers: 2004-229248